Convolutional Neural Networks for Distracted Driver Detection

Rendani Mbuvha, Huijie Wang

KTH Royal Institue of Technology

Abstract. In recent years, Convolutional Neural Networks (CNNs) have become state-of-the-art in Computer Vision. In this work, we apply two transfer learning methods in solving an image classification problem from the Kaggle State Farm Distracted Driver Challenge. We show that using off-the-shelf features from pretrained convolutional neural networks and through fine-turning, high test accuracies of 98.2% and 99.7% are achieved respectively while training on the full set of subjects. However, test accuracy severely decreases to 55.9% as training and testing datasets are split on subjects. As a result, we show the possible overfitting problem which occurs in low-variance datasets and carry out a preliminary discussion on the reasons behind this. Our best fine tuned model, VGG-16 competes relatively well with a Kaggle leaderboard ranking within the top 20% at the time of submission.

Keywords: Convolutional Neural Networks; Alexnet; VGG; Transfer learning; Softmax

1 Introduction

The American Centers for Disease Control and Prevention's (CDC) motor vehicle safety division has reported that one in five car accidents is caused by a distracted driver. This translates to approximately 425,000 injuries and 3,000 fatalities caused by distracted driving every year in the United States. Translated to a global scale this equates to hundreds of thousands of injuries and fatalities annually.In this background, a challenge was step up to help improve these alarming statistics by State Farm, an American Insurer [1] . This challenge involves solving the image classification problem of identifying driver actions based on dashboard camera images.

The application of convolutional neural networks (CNNs) in image classification as demonstrated by [2] achieved significantly high accuracy in Large Scale Visual Recognition Challenge 2012 (ILSVRC2012), this has lead to an explosion of deep learning research and applications. As deeper and larger convolutional nets are being applied, the errors on ILSVRC datasets keep decreasing at an incredible speed, and much research has also endeavoured to find out ways to transfer such models to various applications, ranging from greenfield projects such as self-driving cars to classical problems such as speech recognition. As a result, the authors believe that it is now possible to apply deep convolutional

neural networks in detecting a driver's level of distraction with reasonable accuracy.

Pursuant to the above we use transfer learning methods, which takes advantage of pre-trained networks and modifies them to build classifiers suitable for a specific problem. Two transfer learning approaches are applied in this project and several popular CNNs are tested, including Alexnet, VGG-F, VGG-M and VGG-16. The pretrained models are imported from MatConvnet [3]. Methods in detail including off-the-shelf feature extraction and fine-tuning approaches will be presented in section 4. Background and related work in image classification and transfer learning will be introduced in section 2. And in section 3, the dataset from the competition as well as the library used will be described. This will also include the architectures of the networks we will use. Our experiments and results will be discussed in section 5 and results will be further compared and discussed in section 6. Finally, section 7 will make a conclusion about the project and discuss possible future work.

2 Background

Convolutional Neural Networks (CNNs) were first designed in 1968, motivated by animal visual cortex [4] and designed as a feed-forward artificial neural network. In recent years, CNNs have become a popular method for solving image classification problems since Krizhevsky et al. won Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) with an eight-layer convolutional neural network and a top-5 error of 15.3%, this was almost half of the second best entry's error at the time [2]. This method was applied on Imagenet, a dataset including 1.2 million high-resolution images belonging to 1000 classes. From this success it was found that depth of network significantly influences accuracy. The model was named Alexnet after the first author. Since then, there has been a great amount of research in this area, especially on large-scaled deep convolutional networks, due to their high efficiency , flexibility of structure and ease of training. In the following years of this competition, the accuracy using CNNs keeps increasing and further proved their ability in image classification problems. In 2014, VGGNet reduced top-5 error to 7.3% [5] and GoogleLeNet 6.7% [6] . This two nets go even deeper to reach 16-19 layers and 22 layers respectively. Further more, the Resnet model reduced error to 3.57% in 2015 with the aid of 152 layers.

From this results it can be seen that CNNs are superior to other models/descriptors for image classification problems due to their large and deep structures. However, there are several constraints on the training of CNNs. The first is in the availability of computational resources such as GPU. An typical example is Alexnet. The model has eight learn-able layers was trained with 60 million parameters and 650 neurons, took several days running using cross-GPU parallelization. Secondly, the size of training dataset also has decent influence. Imagenet contains up to 1.3 million images up till now it provides a sufficient representation for various classes of images during training as well as decreases the

probability of overfitting. As a result, it is resource and time consuming to train a CNN from scratch. Thus, a more feasible way to avoid such problem is transfer learning. Several methods are used in transfer learning, such as CNNs fixed feature extractors and fine-tuning the weights of pre-trained networks [7]. Results from Razavian et al. strongly indicate that off-the-shelf features, i.e. features obtained from pre-trained CNNs are efficient to be considered as the primary candidate in most visual recognition tasks [8]. Also Yosinski et al. proved that using transfer features instead of random initialization not only produces better performance (even only fine-tuning weights), but also a boost to generalization [9].

However, although there are some efficiencies using deep learning on classification, there are still no systematic results about the correlations between accuracy and the structure of the CNNs, including filter size, depth, different series of layers. In this project, we apply both off-the-shelf feature extraction and fine-tuning on the distracted driver detection problem and will make discussions about the performance of different approaches.

3 Dataset and Library

3.1 Dataset

The project is taken from the State Farm Distracted Driver Detection [1] challenge. The learning task in this challenge is to classify driver action based on image data. Two datasets are provided on Kaggle: training and tesing dataset. The training dataset are labeled and consists of 22,424 480×680 RGB dashboard camera images, with 26 unique drivers performing various actions. These actions are classified into 10 classes as follows:

- c0: Normal driving
- c1: Texting - right
- c2: Talking on the phone - right
- c3: Texting - left
- c4: Talking on the phone - left
- c5: Operating the radio
- c6: Drinking
- c7: Reaching behind
- c8: Hair and makeup
- c9: Talking to passenger

Figure 1. shows some sample images from this dataset.

On the other hand, Kaggle also provides a testing dataset which contains 79,726 unlabeled images. The two datasets are split on drivers, i.e. a driver can only appear on either training or testing dataset. This test set is used to evaluate models submitted by challenge participants.

As Kaggle doesn't provide naive accuracy for submissions, our experiments mainly depends on the training dataset, but submission results on Kaggle for testing dataset is also included. Pre-processing of this dataset will be explained in Section 3.2.

Fig. 1. Sample Images from the Distracted Driver Dataset

3.2 Data Pre-processing

Since we focus mainly on feature extraction and fine-tuning pretrained models some preprocessing steps were taken before the image data was parsed through the networks. This in the main involved resizing the images and normalizing the image input channels with the means that were obtained when the models were initially trained.

We further split the 'training' dataset into training, validation and test sets. This split was done in ratios of 75%:20%:5% respectively resulting training, validation and test sets of sizes 16819:4484:1121.

In addition, we also do experiments on splitting drivers. In which images for 5 drivers are set out as validation data, accounting for 3,837 images totally. For comparison, the rest of the data is also split into two parts, 14,750 for training and 3,837 randomly selected images from the rest 21 drivers as a second validation dataset.

3.3 Matcovnet

The convolutional neural networks in this project were built using the Matlab based Matconvnet library[10]. This library was developed by the Visual Geometry Group(VGG) at Oxford University. In this library various pretained models are readily available, including Alexnet, VGG-F,VGG-M and VGG-16 which are utilized in this project.

4 Methods

In this project, we mainly use transfer learning methods to solve the distracted driver detection problem. Pre-trained Alexnet, VGG-fast,VGG-medium

and VGG-16 are introduced from Matconvnet [3] and two approaches, off-the-shelf and fine-tuning of weights, are applied to this problem. The architectures of these models are discussed in this section.

4.1 Off-the-shelf feature Extraction from Pretrained CNNs

One of the fundamental revelations that has come with the rise of convolutional nets has been the realisation that representations learnt from training a net for one task can be transferred 'off-the-shelf' to another task and still give competitive performance on the new task. [8] show this by performing different experiments on various object recognition tasks using features extracted from the first fully connected layer of the *overfeat* network. Through these experiments they show that using the above features and a linear SVM classifier outperforms most of state-of-the-art descriptors.

In our feature extraction experiment we use the representations from the first fully connected layer of Alexnet [2]. Using these representations which are of 4096 dimensions we train a Softmax classifier with 10 output classes. We perform the Softmax training using the neural network toolkit in Matlab with 40 ephocs.

The Softmax Classifier is a generalisation of logistic regression for multi-class classification defined by the function:

$$softmax(x, k) = \frac{e^{x^T w_k}}{\sum_{k=1}^{K} e^{x^T w_k}}$$

for feature vector x and a target class k. The predicted class is then calculated using the maximum posterior probability.

In training this network we aim to learn the weights w_k that minimize the cross-entropy between the model predictions and the ground truth. The output of the softmax is posterior probability of the example belonging to class k given the feature vector x.

4.2 Fine-tuning the CNNs

We also use fine-tuning method to train our models. In this method, the intial weights of every layer but last layer are imported from pre-trained models and 20 epochs of learning are applied in order to fine-tune the weights to the current problem. The detailed structures of the models are discussed in this section.

4.3 Architectures

4.3.1 Alexnet Alexnet is a large, deep convolutional neural network which won Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) [2]. The structure of Alexnet includes five convolutional layers and three fully-connected layers. The convolutional layers provide good representation of local features of image while fully-connected layers mainly learn more general features. As a result, the size of image is binded due to the fully-connected layers, and here

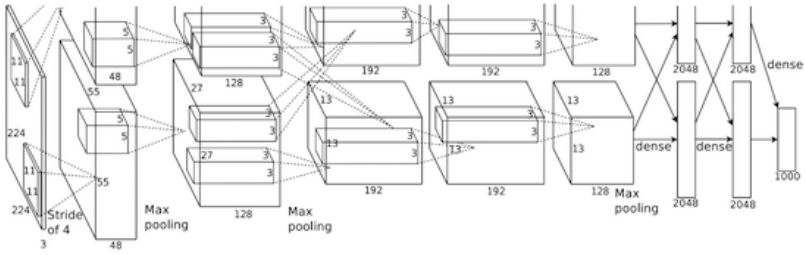


Fig. 2. Architecture of Alexnet. Figure from [2]

in our project image size is down sampled to $227 \times 227 \times 3$. The structure of Alexnet is shown in Figure 2.

As it can be seen, each of the linear layers are followed by several normalization layers. In the first three convolutional layers, ReLU, response-normalization and max-pooling layers are attached. However, the forth convolutional is only followed by a ReLU layer and the fifth only followed by a ReLU layer and a Pooling layer. The first two fully connected layers are attached with a ReLU layer and a Dropout layer, with a $dropoutrate = 0.5$. The dropout layer is used to set outputs of neurons of the attached hidden layer to zero with a probability of 0.5. This method not only reduce the computational cost of deep neural network, but also the complex co-adaption of neurons. As a result, it is an important method to reduce overfitting problem. The network is fed into a softmax loss regression for stochastic gradient descent.

In our project, Alexnet is built using the $SimpleNN$ function of MatConvnet. In order to reduce the training time, the code is rewritten and introduced with parameters of pre-trained model as initial weights in our network, except for the last layer, where random weights are created during initialization. This proved to be more efficient and requires less epochs to converge. The model is trained using a batch size of 128, following [2], and a learning rate of 0.001, which is tested to be efficient according to our trials. The size of the output of the last fully convolutional layer, which acts as a prediction layer, is changed from 1000 to 10 corresponding to our prediction classes. On the other hand, restricted by computational resources, dense connectivity is used between convolutional layers instead of the sparse connections used in [2].

4.3.2 VGG Despite the impressive results achieved by convolutional neural networks recent years, there is still the non-trivial problem of evaluating how different structures of CNN perform relative to each other. [11] conducted a rigorous evaluation on the new techniques, and explored different deep architectures and comparing them on a common ground. Three models, Fast (VGG-F), Medium (VGG-M) and Slow (VGG-S) are carried out, in which VGG-F is very similar to Alexnet. On the other hand, VGG-M also has a similar structure but stride sizes and receptive fields are modified. The first convolutional layer's stride size is decreased to 2 (considering that this size is proved to be more suitable

for Imagenet dataset), while it is increased to 2 in the second layer to limit computation into a reasonable time. On the other hand, the filter size of the first layer is reduced to 7×7 in order to reduce the receptive field. In our project, VGG-M is implemented using the same method of initialization as Alexnet, as well as the same parameters of batch size and learning rate. The structures are shown as Figure 3.

Arch.	conv1	conv2	conv3	conv4	conv5	full6	full7	full8
CNN-F	64x11x11 st. 4, pad 0 LRN, x2 pool	256x5x5 st. 1, pad 2 LRN, x2 pool	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 x2 pool	4096 drop-out	4096 drop-out	1000 soft-max
CNN-M	96x7x7 st. 2, pad 0 LRN, x2 pool	256x5x5 st. 2, pad 1 LRN, x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x2 pool	4096 drop-out	4096 drop-out	1000 soft-max
CNN-S	96x7x7 st. 2, pad 0 LRN, x3 pool	256x5x5 st. 1, pad 1 x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x3 pool	4096 drop-out	4096 drop-out	1000 soft-max

Fig. 3. Architectures of VGG-16. Figure from [11]

In our project, VGG-F and VGG-M are tested. The depth of VGG was increased to 16-19 layers during ILSVRC2014 [5]. Limited by computational resources, we only introduce VGG with 16 layers (VGG-16). The structure of VGG-16 is shown as Figure 4. It consists of 13 convolutional layers and 3 full-connected layers. Furthermore, despite the depth, it also differs from Alexnet in the size of filters are fixed to 3×3 and continuous linear layers.

Arch.	1	2	3	4	5	6	7	8	9
VGG-16	conv3-64	conv3-128	conv3-256	conv3-512	conv3-512	fc-4096	fc-4096	fc-1000	softmax
	conv3-64	conv3-128	conv3-256	conv3-512	conv3-512				
			conv3-256	conv3-512	conv3-512				
	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU		
	x2 pool	x2 pool	x2 pool	x2 pool	x2 pool	Dropout	Dropout		

Fig. 4. Architectures of VGG nets. Figure elicited from [5]

5 Experiments and Results

5.1 Off-the-shelf features

As described in section 4 our experiments include testing the accuracy of a softmax classifier trained on features extracted from the first fully connected layer of Alexnet. The softmax classifier was trained on 16819 training examples and tested on 1121 unseen examples. The testing performance of the softmax

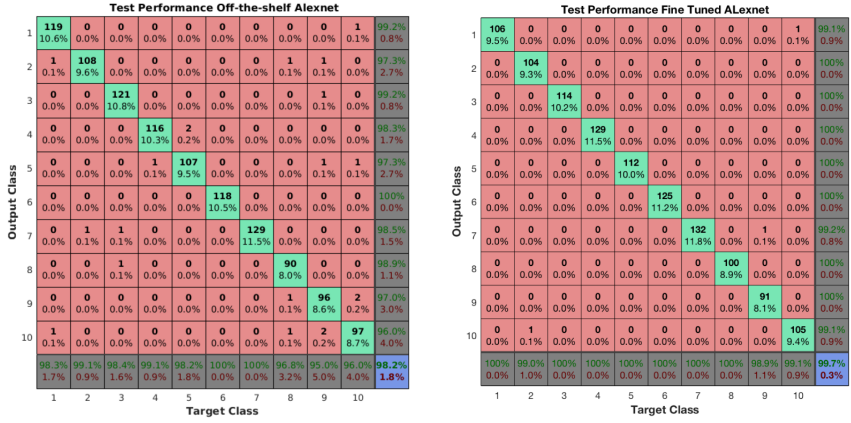


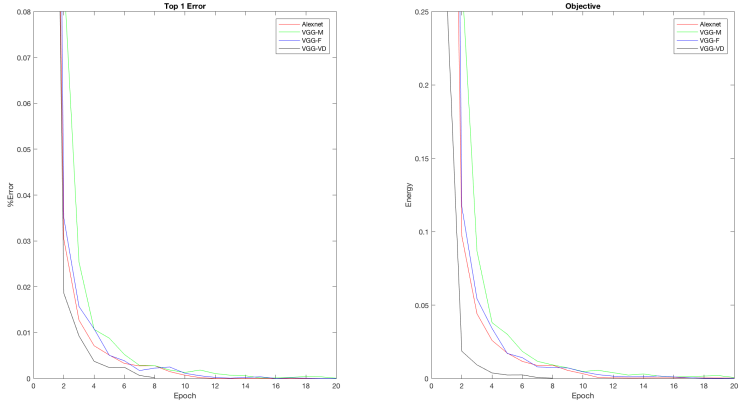
Fig. 5. Comparison of the confusion matrices for the softmax classifier and a fine turned Alexnet

classifer was compared to that of an Alexnet that was fine-tuned (which will be discussed in detail in section 5.2) on the same training examples.

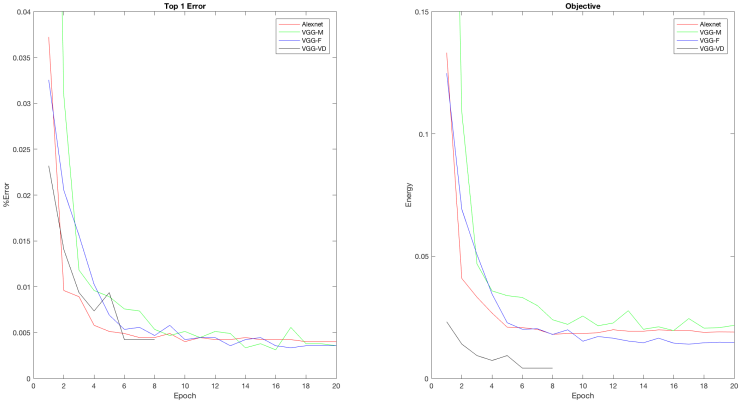
Our results are summarized by the confusion matrices in Figure 5.1. We obtain an overall classification accuracies of **98.2%** with the off-the-shelf classifier and **99.7%** with the fine tuned CNN on the testing set. However, it should be mentioned that we achieved 100% accuracy on training data using this approach.

5.2 Fine-tuning on multiple models

As shown in the previous section, it is possible that due to the differences between distracted driver detection dataset and Imagenet, extracting features directly using pre-trained Imagenet-based model may not have as good performance as fine-tuning. As a result, in this section we focus on fine-tuning based on Alexnet, VGG-F, VGG-M and VGG-16. It is shown in section 4.3 that VGG-F has a very similar structure with Alexnet implemented in our project, despite minor differences of filter depth and numbers. Both of the models have advantage on training speed from the large stride size of the first convolutional layer. On the other hand, VGG-M decreases the receptive fields by using smaller filters. While the stride size of the first two layers are also adjusted. The last CNN we used, VGG-16 has a relatively different structure with the previous three. The depth of the net allows it to dig further into image features. And other main features of VGG-16 lies on fixed filter size (3×3) and continuous convolutional layers.



(a) Top-1 error rate and energy on training dataset



(b) Top-1 error rate and energy on validation dataset

Fig. 6. Results of fine-tuning

In our project, the four CNNs are all slightly modified by changing the prediction neurons size from 1000 to 10 in order to fit our data classed. Ideally we tend to run 20 epochs of training, however, limited by the time, we are only able complete 8 epochs of training on VGG-16 due to its depth and high computational complexity. Each epoch the nets are passed over the entire training set, which is 75% of the whole dataset broken down into batches of 128. Learning rate is set to 0.001 according to our trials. The results of the experiments are shown as Figure 6.

In the plots the training of VGG-16 stops after 8 epochs. Alexnet and VGG-F has very similar performance on training data. The decrease of error and energy of VGG-M are slightly slower than Alexnet and VGG-F but also converges after 15 epoches. However, it is shown that on training dataset the former two CNNs has a more stable performance. On the other hand, VGG-16 outperforms the other three CNNs both on training and validation.

5.3 Split on subjects

Until now the experiments are all done without spliting the training and test sets on drivers. In this part we set out 5 drivers accounting for 3,837 images as validation set ,and for comparison, another 3,837 images randomly selected from the other drivers are also set out for testing. Alexnet was trained based on the new training dataset, and the result of the experiment can be seen from Figure 7.

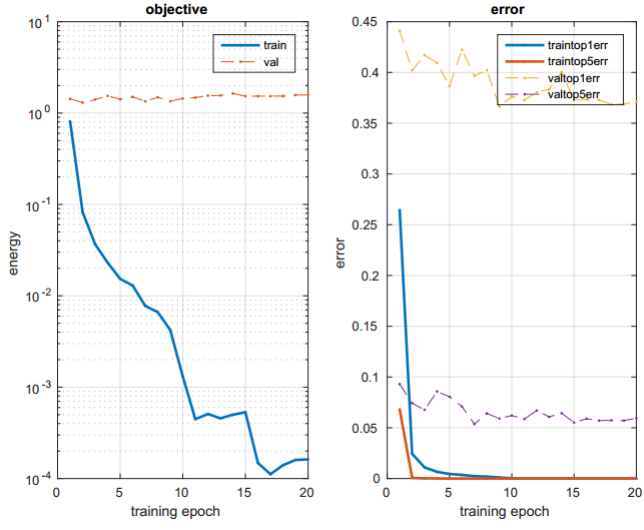


Fig. 7. Alexnet trained on datasets split by drivers

In the figure, it is shown that, though the training error and loss keeps decreasing, those of validation sets remains high and the error (we mainly focus on Top-1 error) reduces slow and with high fluctuation. The accuracies described by confusion matrices are shown as Figure 8.

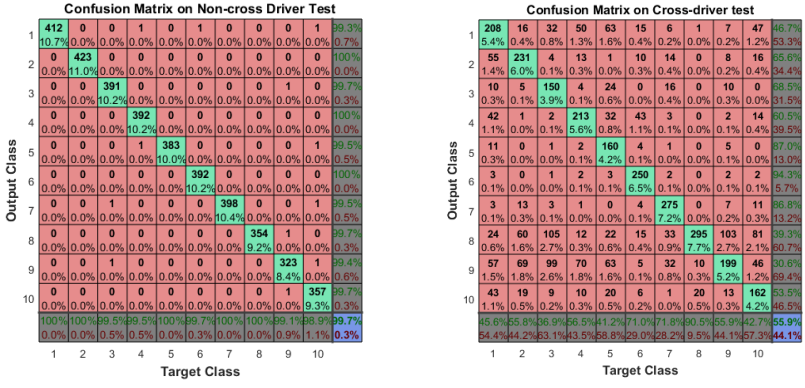


Fig. 8. Comparison of the confusion matrices for non-cross driver result and cross-driver result

As can be seen, the accuracy decreases on the cross-driver testset, severely dropping from 99.7% to only 55.9%. According to the matrices, the network has a higher probability to classify images into class 8 and 9, i.e. class "Reaching behind and class "Hair and makeup, which receive only around 30% of precision among those images being classified into the two classes. Thus, though on class 8 the matrix shows a higher accuracy around 90%, the overall accuracy remains quite low.

As a conclusion, the CNN does not show a efficient ability of transferring driver/subject invariance. The original parameters from the pretrained Alexnet already account for 90% of the accuracy and further training may in overfitting.

5.4 Kaggle scores

In addition to the 22424 training images, Kaggle also provides 79,726 unlabeled images for challenge participants to be evaluated on. Challenge participants submit a CSV file containing softmax probabilities for each image based on their model[1]. The submissions are evaluated using a multiclass logloss defined as follows :

$$logloss = \frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij}\log(p_{ij})$$

for N images with M classes where y_{ij} is the correct class of image i and p_{ij} is the predicted softmax probability.

We submitted the predicted probabilities from our models to Kaggle, the results were as shown in Figure 9.

Model	logloss	Rank*
Off-the-shelf Features	6.27736	950+
Fine tuned Alexnet	1.17977	343
Fine tuned VGG-16	0.71759	184*

Fig. 9. Kaggle Submission results showing logloss and Rankings. The rankings are as at the time of submission out of 963 submissions

From figure 9 we see that VGG-16 outperforms the other two models even with only 8 epochs of training results in a ranking within the top 20% of submissions. Using off-the-shelf features seems to not work well when data is totally unseen. While using a fined Alexnet achieves slightly better performance it still highlights inability to generalise well.

6 Discussion

Our results show that although using off-the-shelf features provides high test accuracies, it can be further enhanced by fine-tuning a pretrained network. This is consistent with guidance from [7] which suggests that if a dataset significantly different from the original dataset used to train the network, fine-tuning can assist increasing specificity to the new task. The result is further proved by the scores returned from Kaggle, where multiclass loss on off-the-shelf method is almost six times of fine-tuning method.

The problem can be discussed from two aspects. Firstly, since our dashboard camera dataset is set within a more specific environment than the very diverse Imagenet dataset, and the training dataset is much smaller comparing to Imagenet dataset, training with only our dataset using deep neural networks could cause problem of overfitting. The problem about how much deep CNNs overfit is also discussed when Alexnet was carried out. One hypothesis is that "ImageNet is large enough to support training an AlexNet architecture without excessive overfitting" [9]. On the contrary, using off-the-shelf features extracted directly from Imagenet-based CNNs may suffer the problem of lack specific representation of the dataset. Thus, even we achieved 100% accuracy after training with softmax, the testing result was worse. The phenomenon implies that the features are not fully representable for the dataset and causes overfitting after learning. As a result, using fine-tuning as a transfer learning approach could be more suitable for our dataset. It could also be beneficial to try out representations from other layers of the network.

In comparing the results obtained by fine-tuning of four different pretrianed networks. Only taking into account the first eight epochs, we observe that VGG-16 largely outperforms the other three CNNs minimizing softmax loss both on training and validation dataset. It also shows a faster decrease on training and validation Top-1 errors. The Kaggle results further proves this observation, where VGG-16 decreases the multiclass loss by about 0.4 against from the Alexnet result. It is obvious that a deeper network could have advantage against smaller CNNs, which is in accordance with the results from [5] and [6]. The conclusion may be further tested given more computational resources.

Among the three eight-layer CNNs, Alexnet shows the most stable performance. Moreover, it shows that Alexnet tends to have a more efficient decrease of error rate and energy during the first several epochs but VGG-F performs better with more epochs on validation dataset. The result is quite interesting taking into account that in our project Alexnet and VGG-F have a very similar structure. The reason for the difference remains unknown and may need further experiments to examine whether it is merely a random result.

Meanwhile, VGG-M also has a similar results after several epochs, it has the least loss decrease rate in the initial ephocs. The phenomenon may arise from a smaller receptive field. As it can be seen, the images in the dataset provided by State Farm are highly normalized. The size of the images, the background, the position of the subject are all very consistent. As a result, a global representation of the images may have more influence on the result.

However, although these CNNs are proved efficient within the training and testing on the same driver sets, the accuracy severely decreases when a cross-driver test is introduced into the experiments. On our small-scale test by setting out 5 drivers as validation, the accuracy drops to 55.9%. And according to the scores returned from Kaggle, where all the images are tested on un-seen drivers in the training set, the best multiloss is performed by VGG-16, of 0.71759. That is to say, approximately 20% may be correctly classified on average. As a result, the CNNs have an excellent performance on training dataset but poor transfer to unseen subjects.

Thus, we do find generalization on training dataset is a serious problem in fine-tuning method. An assumption about the result is that merely 26 subjects in the training dataset is not large enough for representing the features of the classes and results in overfitting (similar assumption to overfitting problem of Alexnet).

7 Conclusion

In this work we have applied two transfer learning methods to Distracted Driver Detection problem. According to the results, we have seen that off-the-shelf features from Alexnet can be further improved by fine-tuning based on pretrained networks. Both off-the-shelf method and fine-tuning method provide reasonably high test accuracies within the dataset of same subjects, but perform relatively worse when the training and testing datasets are split by drivers.

The problem of generalization and overfitting on this artificial dataset remains unsolved in our work.

We also compared the results of fine-tuning on different network architectures. As a result, we found that VGG-16 displayed superior performance in terms of both training and validation error even with reduced number of ephocs. Meanwhile, the results of other three CNNs are similar but Alexnet shows a more steady trend of decrease of error. It shows that going deeper into layers does have considerable influence on accuracy.

Thus, in future experiments, several approaches could be applied as candidates of ways solving the problems. Firstly, going deeper is a good try in order to represent more levels of features in the dataset. Secondly, as merely classification has a limited performance in this problem, some other techniques, such as key point detection, may be useful. Thirdly, the dataset provided by State Farm is artificial. We can see that only 26 subjects are included but repeated with different postures thousands of times. Also the background and subject positions are similar. It is a reasonable assumption that the dataset may cause overfitting on the certain subjects. As a result, using a more general dataset, with various subjects performing different postures may be more ideal for training and generalization.

Limited by time and computational resources, only few experiments were completed and for a more systematic analysis of the results, more training and validation should be done on different structures as well as different parameters. For example, we only the accuracies of the four CNNs are only based on one run, thus it is possible to be a random result, especially on the comparison of Alexnet, VGG-F and VGG-M. Also more tests could be done by extracting different layers on the off-the-shell approach.

As a future addition to this work one also can consider varying some of the vast number of hyperprameters that are involved when training CNNs. These include but are not limited to layer-wise learning rates, weight initialisations and network architectures. Also the complexity of the networks could be considered in context of their performance thus highlighting effects of over-parameterizing and over-learning. It is also possible to test with real-time monitors in cars using 3-D convolutioanal neural networks.

Acknowledgements

The computations in this work were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at PDC Centre for High Performance Computing (PDC-HPC)

References

1. State farm distracted driver detection, 2016.
2. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

3. Pre-trained models, 2016.
4. David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.
5. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
6. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
7. Transfer learning, 2016.
8. Ali Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.
9. Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.
10. Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 689–692. ACM, 2015.
11. K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.